behaviac 性能分析

腾讯游戏
Tencent Games

https://github.com/TencentOpen/behaviac

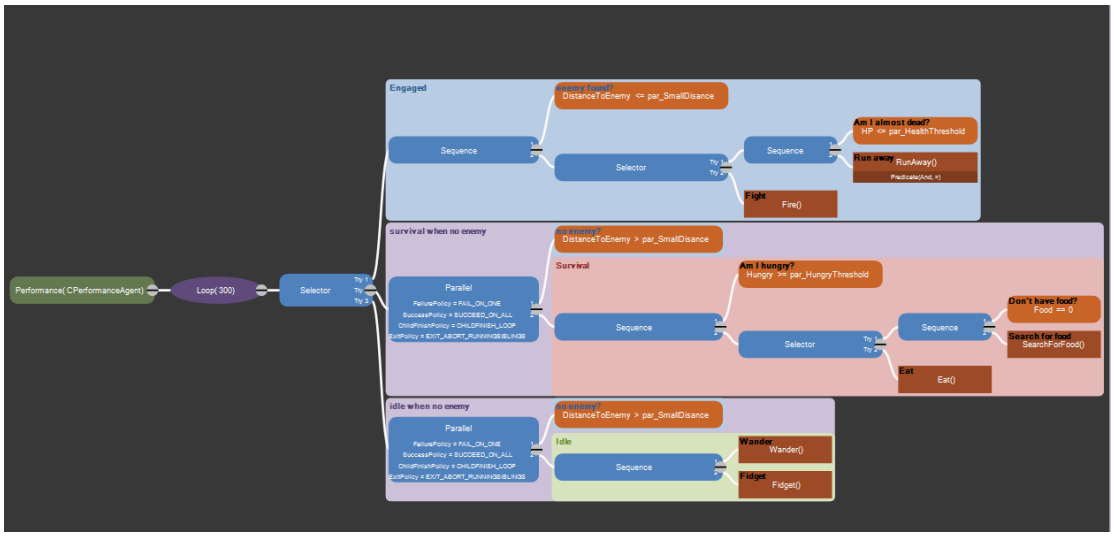
目录

1 概述

除了易用性，性能也是设计和实现 behaviac 时候重点考虑的因素。

对于运行状态为‘Running’的节点，在接下来的执行过程中，直接执行该节点，而不需要从树的开始从新执行。此外，行为树可以以源码（C++、C#）的形式导出，从而以最为高效的形式执行。
在 unity 版本中，在执行行为树的过程中，不需要从堆分配内存（Heap allocation）（用导出的 C#文件执行的时候没有 Heap allocation，用 xml、bson 的时候还是有可能需要 Heap allocation 的）。
主要基于以上优化，behaviac 的执行过程极为高效，下面将分别按 C#，C++等来统计执行消耗。

整个统计过程所用的行为树如下图：

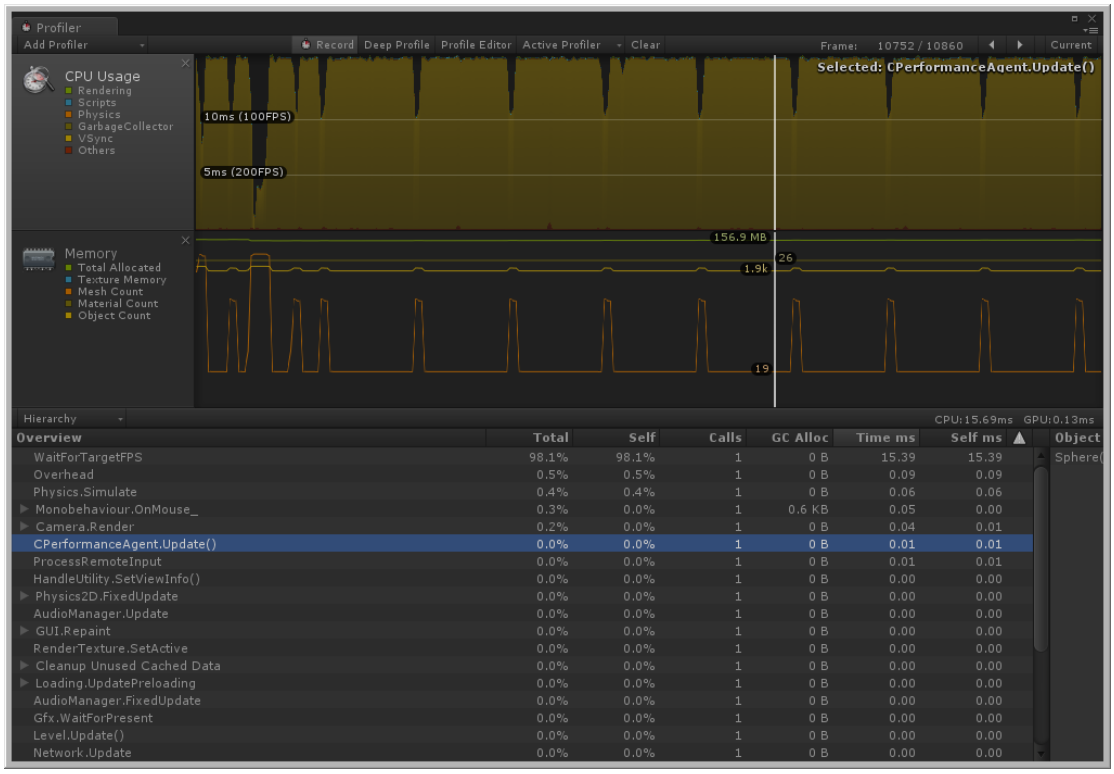


每个 Agent 上运行如上图所示的一个行为树。

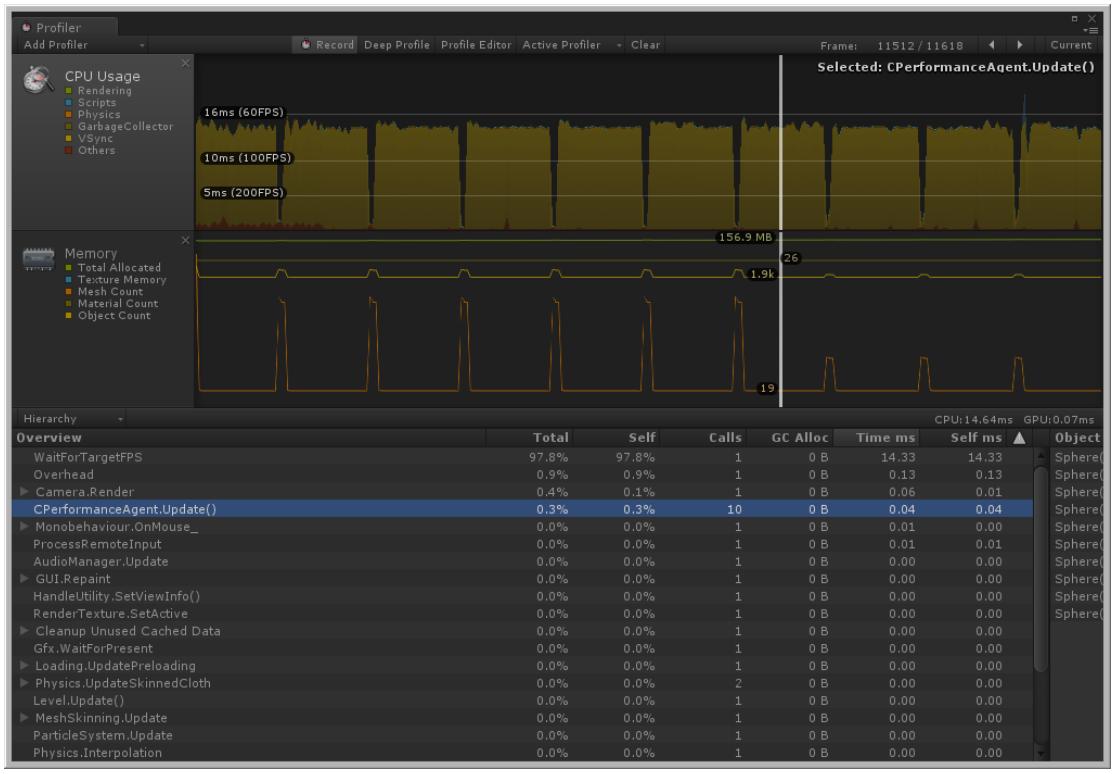
2 运行性能的统计

2.1 PC， i7 2.93GHz, 8G RAM

1 Agent

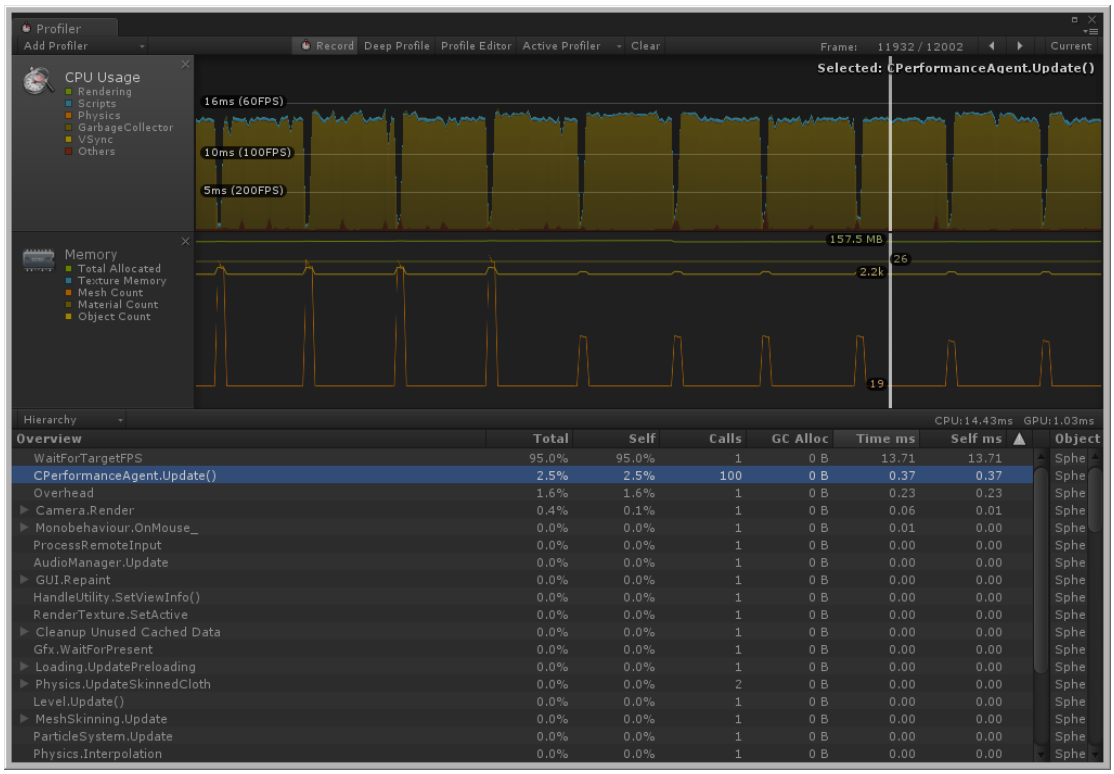


10 Agents

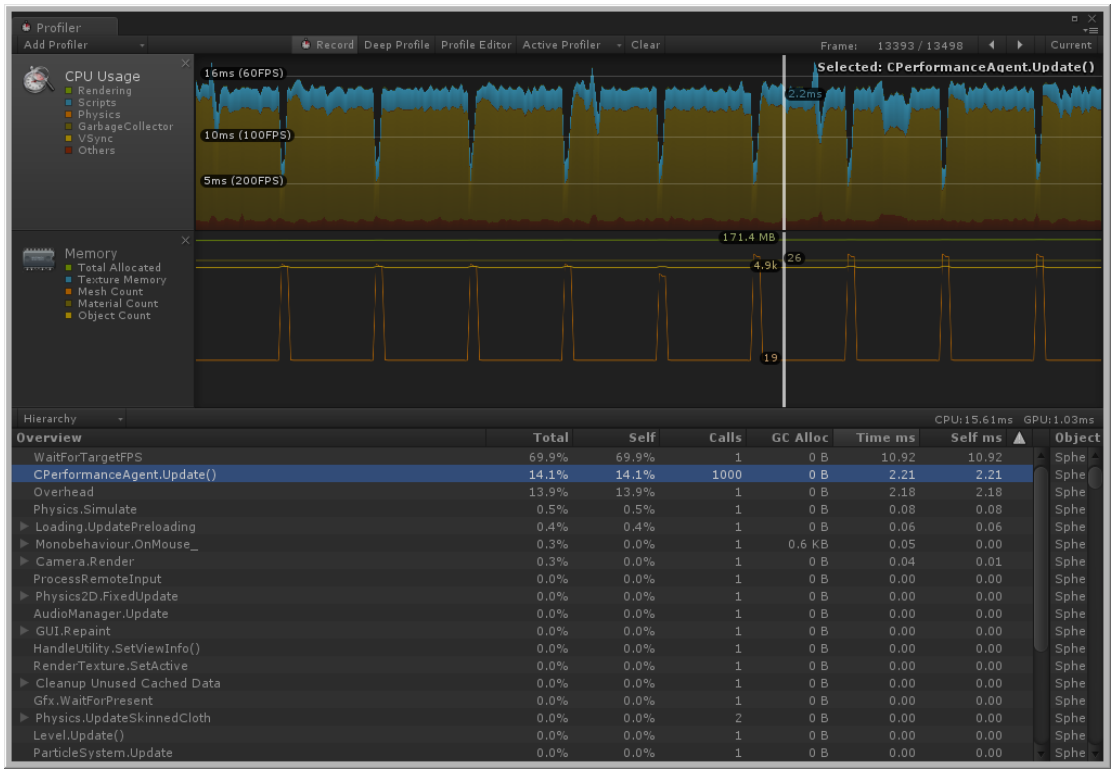


https://github.com/TencentOpen/behaviac

100 Agents

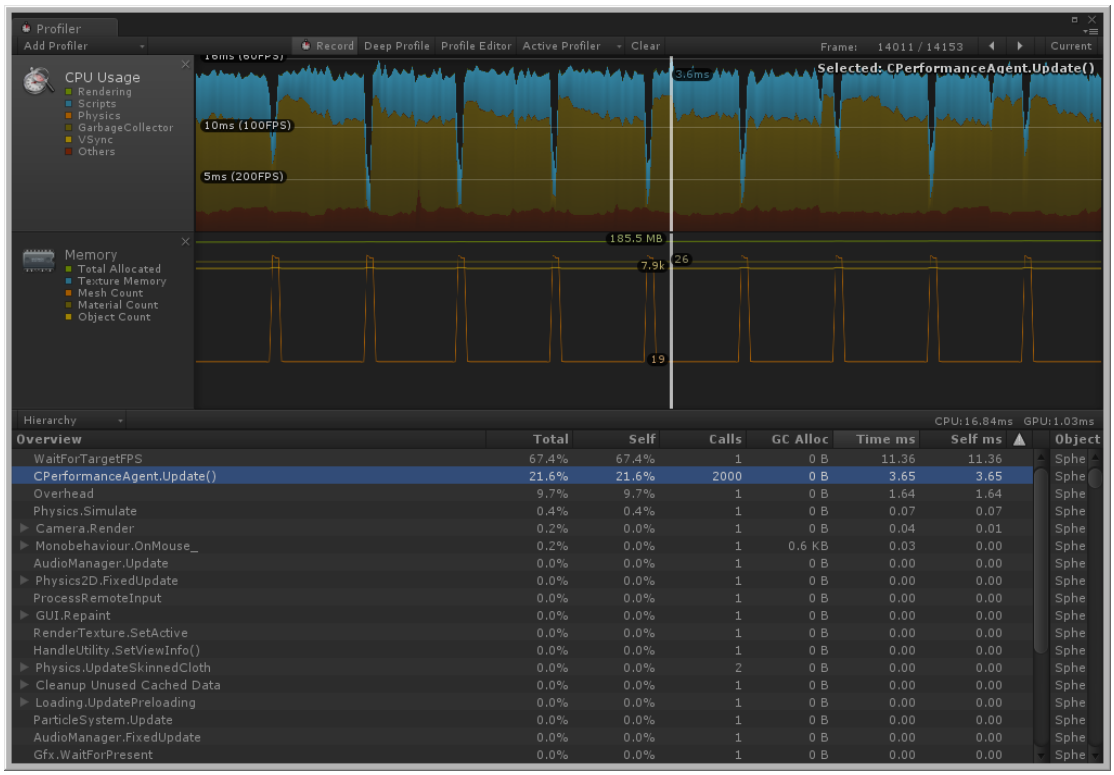


1000 Agents



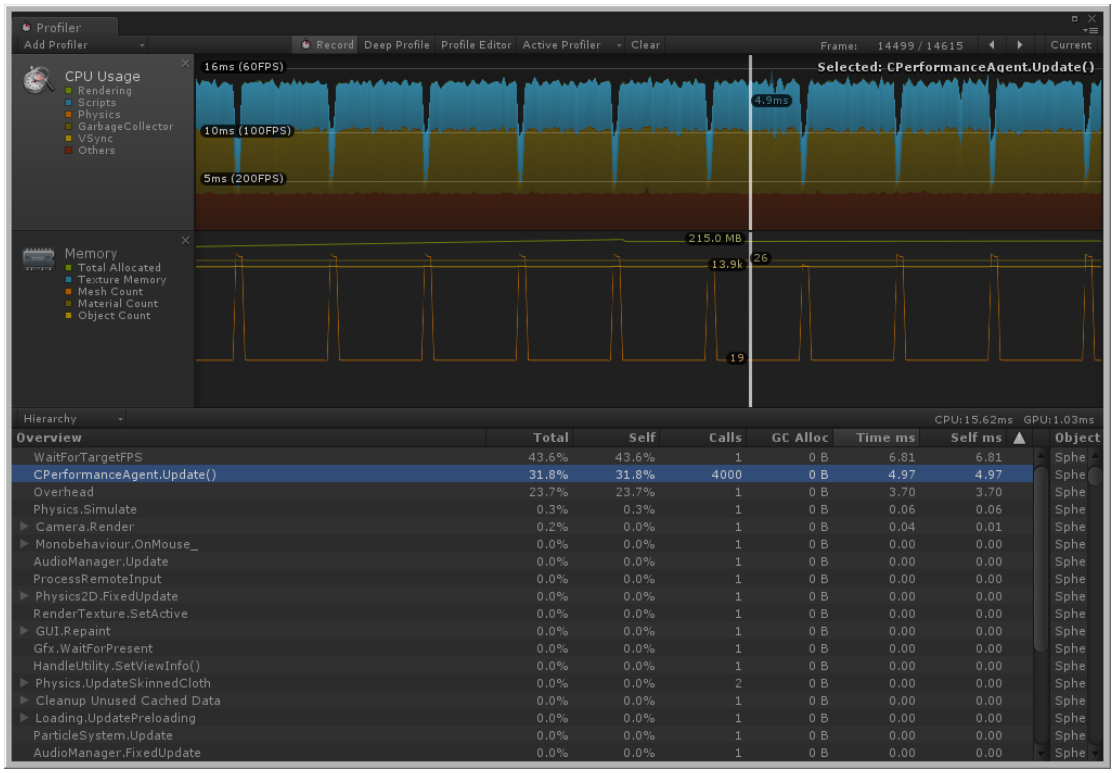
https://github.com/TencentOpen/behaviac

2000 Agents



4000 Agents

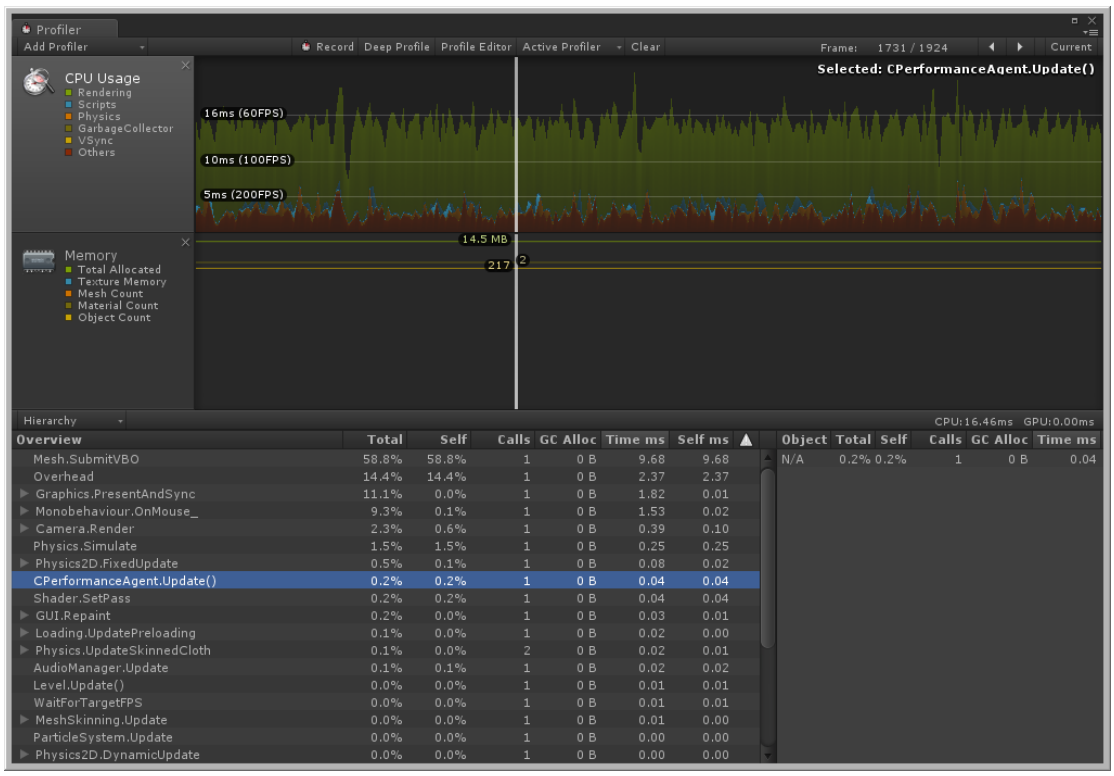
https://github.com/TencentOpen/behaviac



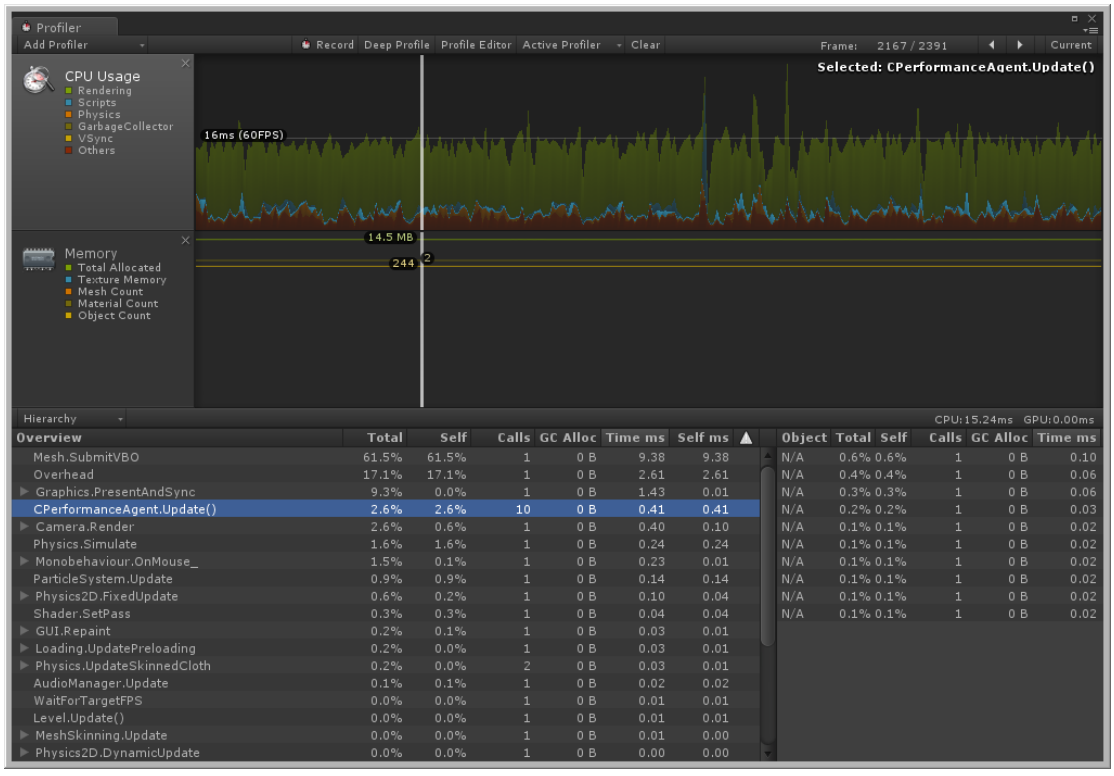
2.2 Android， MX5Q Dual Core 1.4GHz， 1G RAM

1 Agent

https://github.com/TencentOpen/behaviac

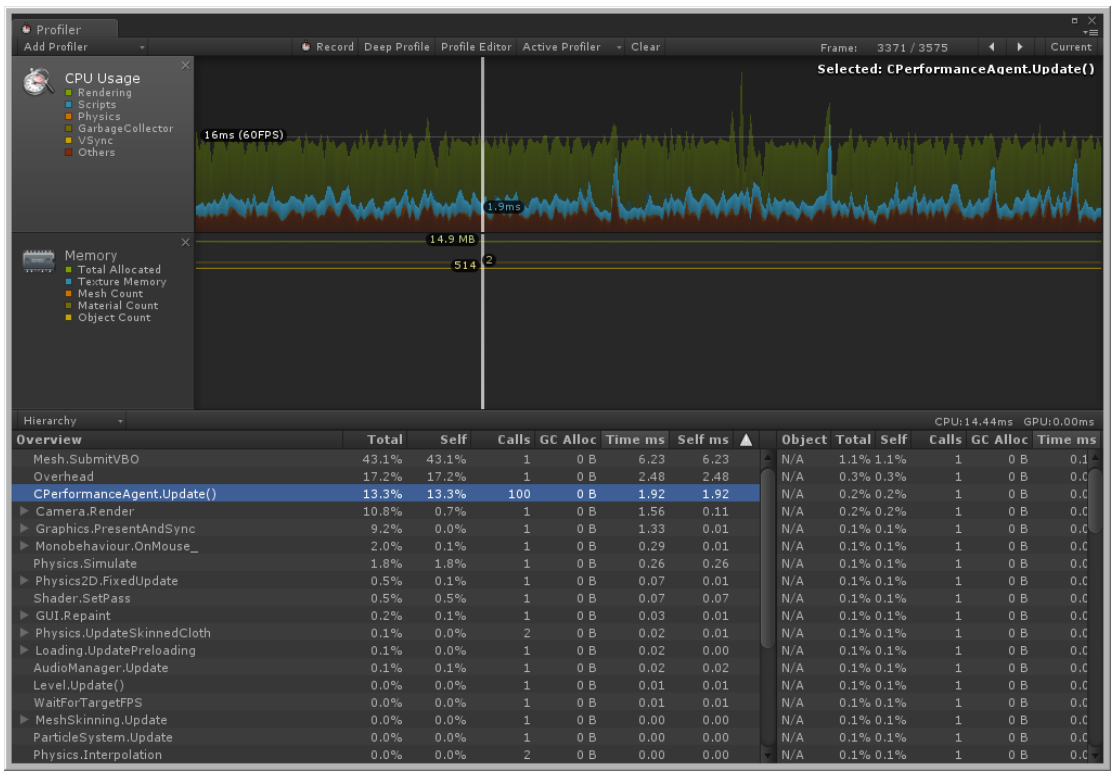


10 Agents

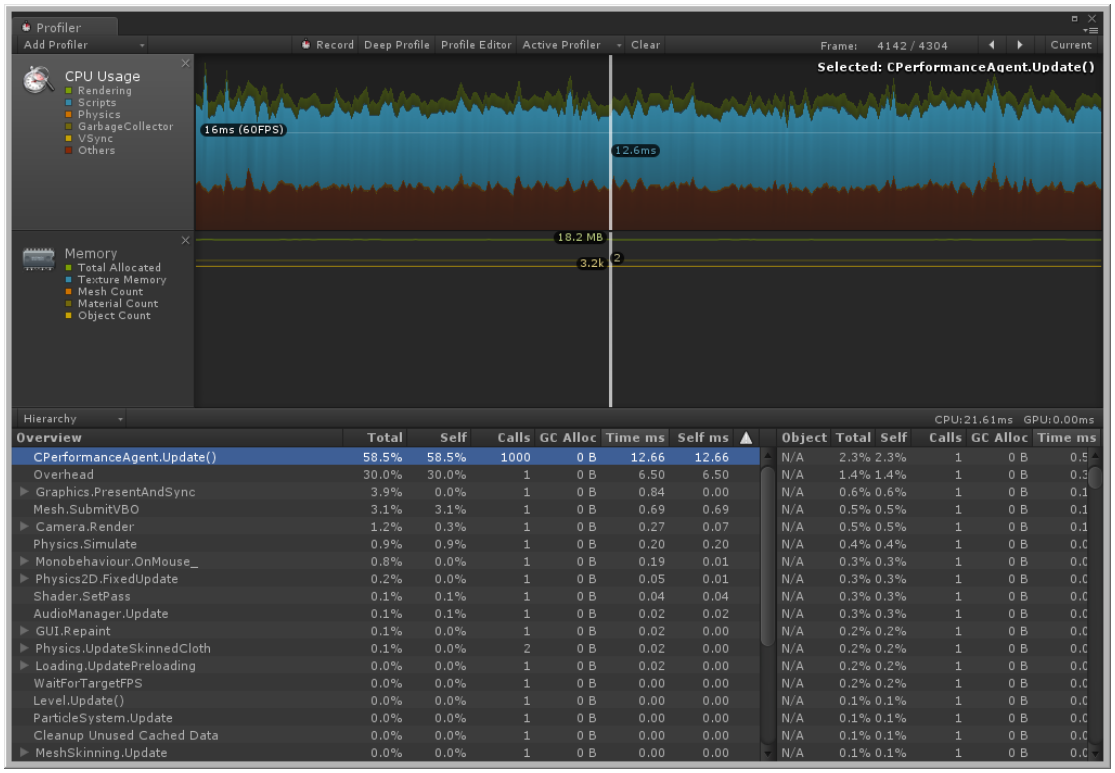


100 Agents

https://github.com/TencentOpen/behaviac

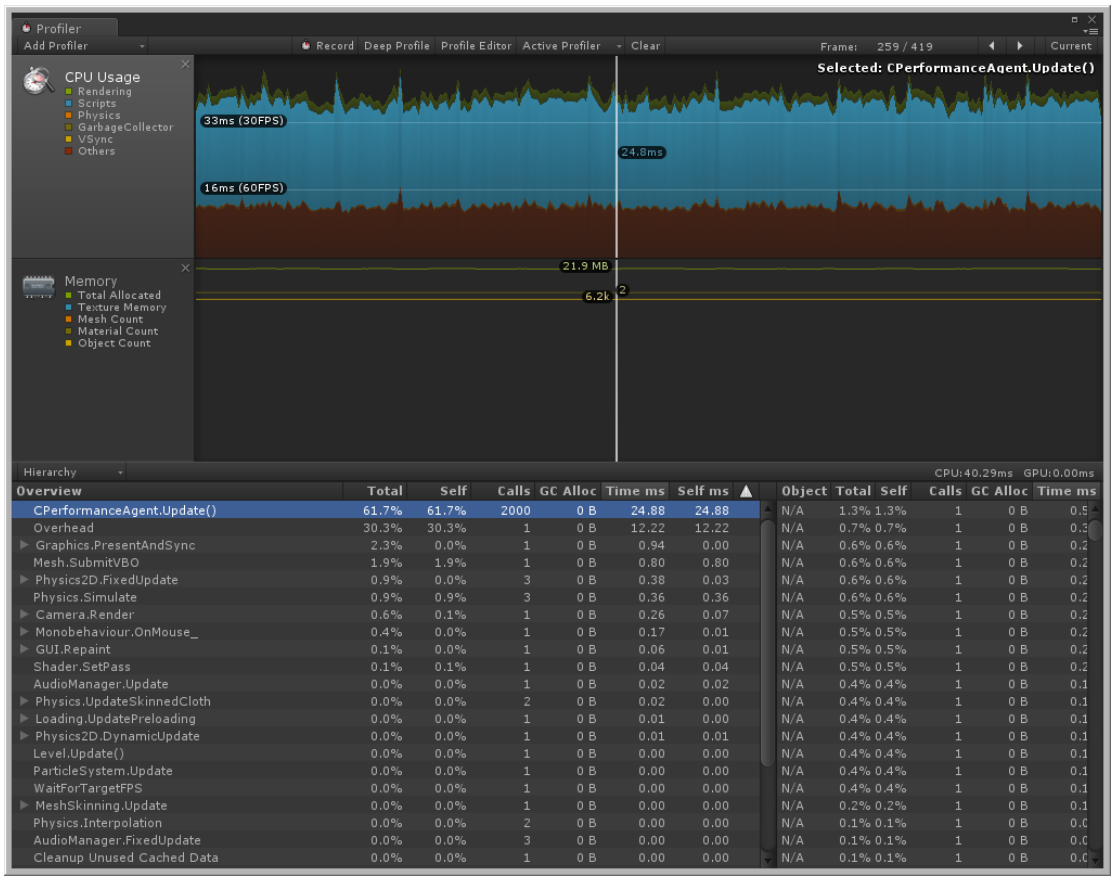


1000 Agents

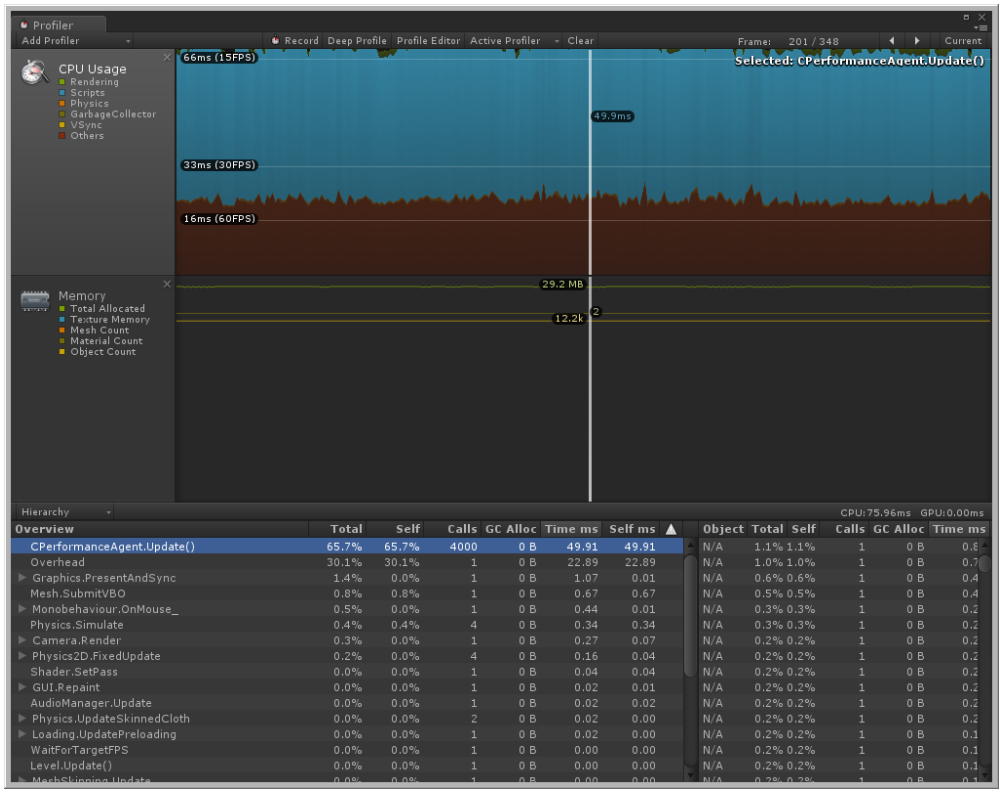


https://github.com/TencentOpen/behaviac

2000 Agents



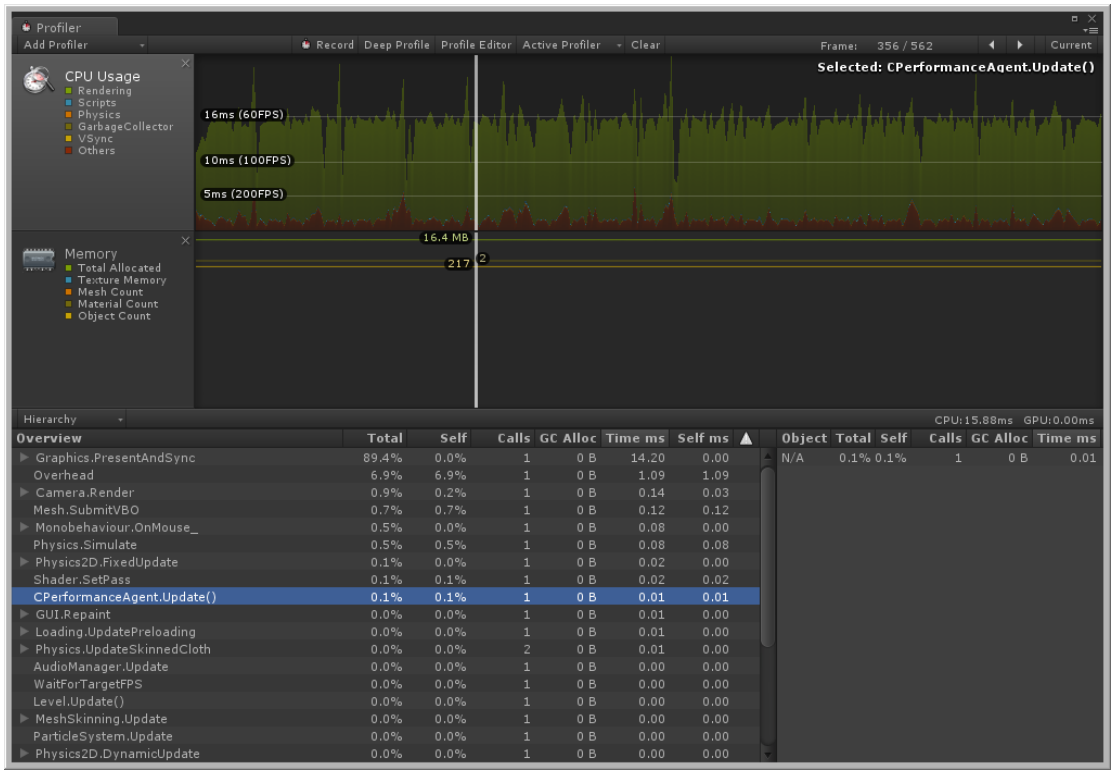
4000 Agents



https://github.com/TencentOpen/behaviac

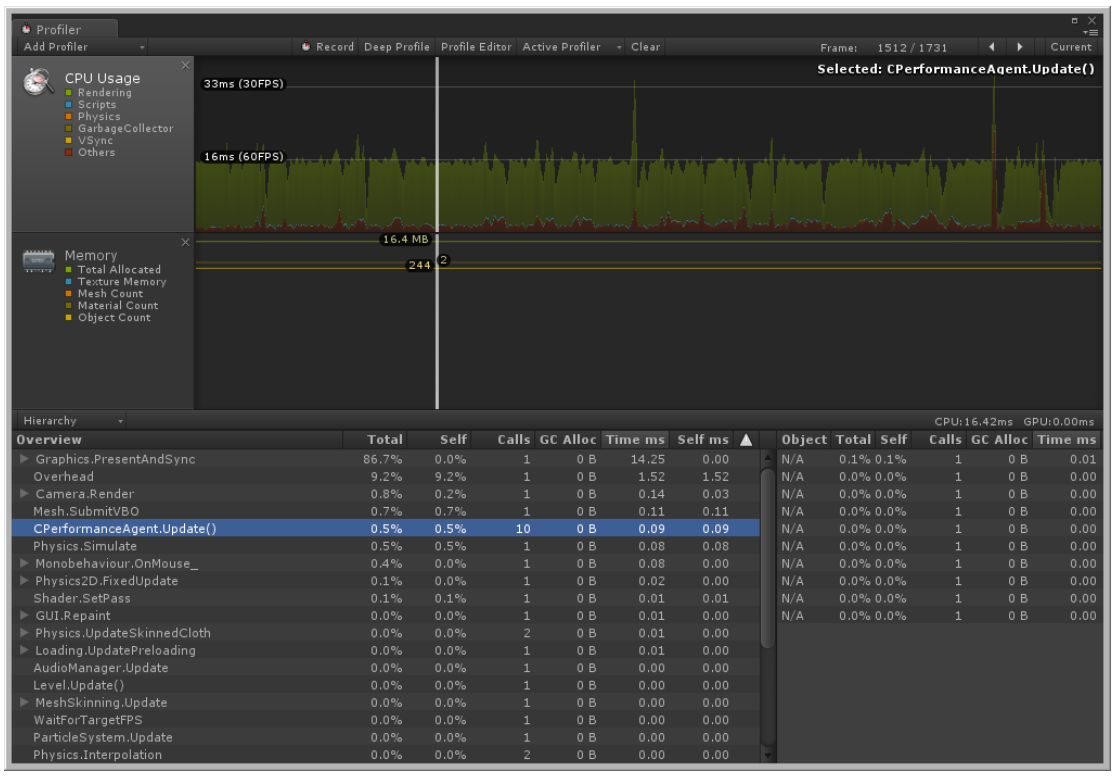
2.3 Android， MSM8974 Quad Core 2.2GHz， 2G RAM

1 Agent

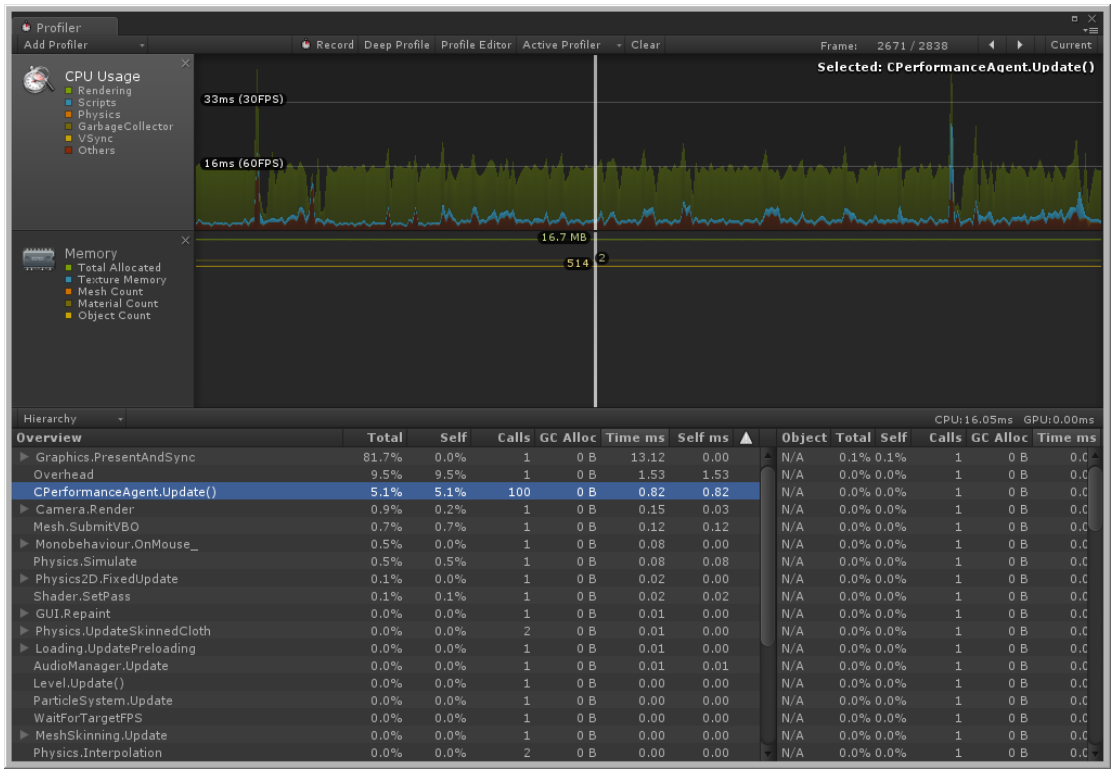


10 Agents

https://github.com/TencentOpen/behaviac

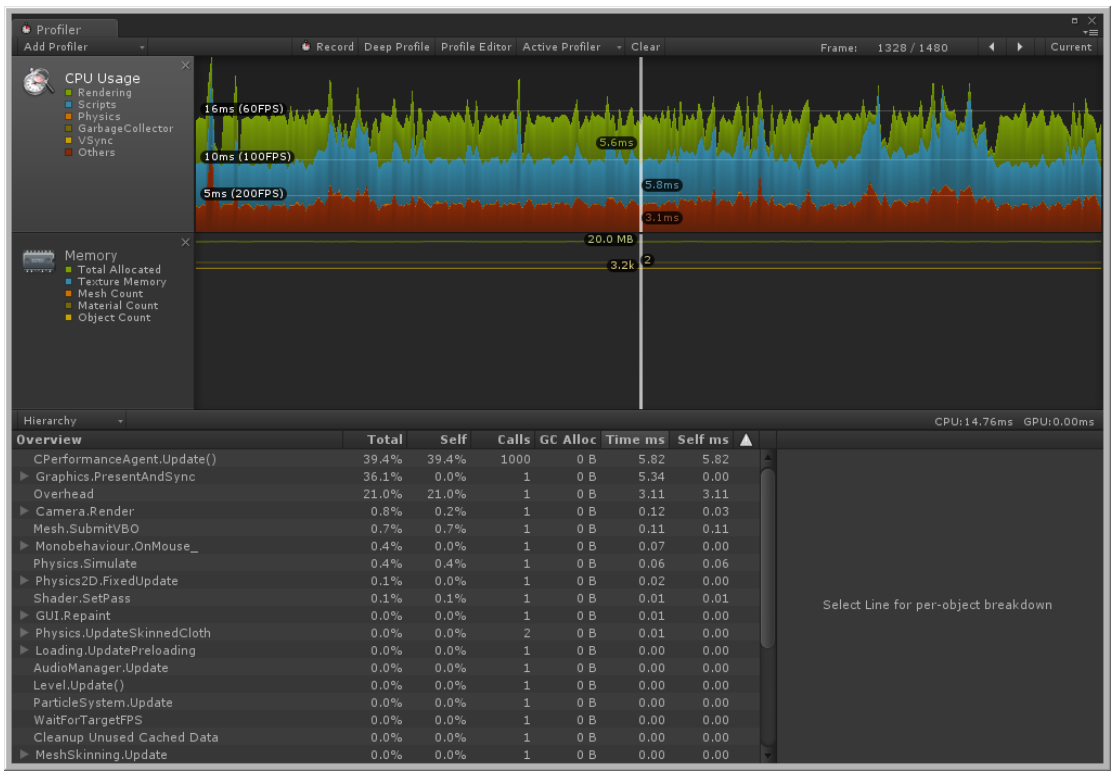


100 Agents

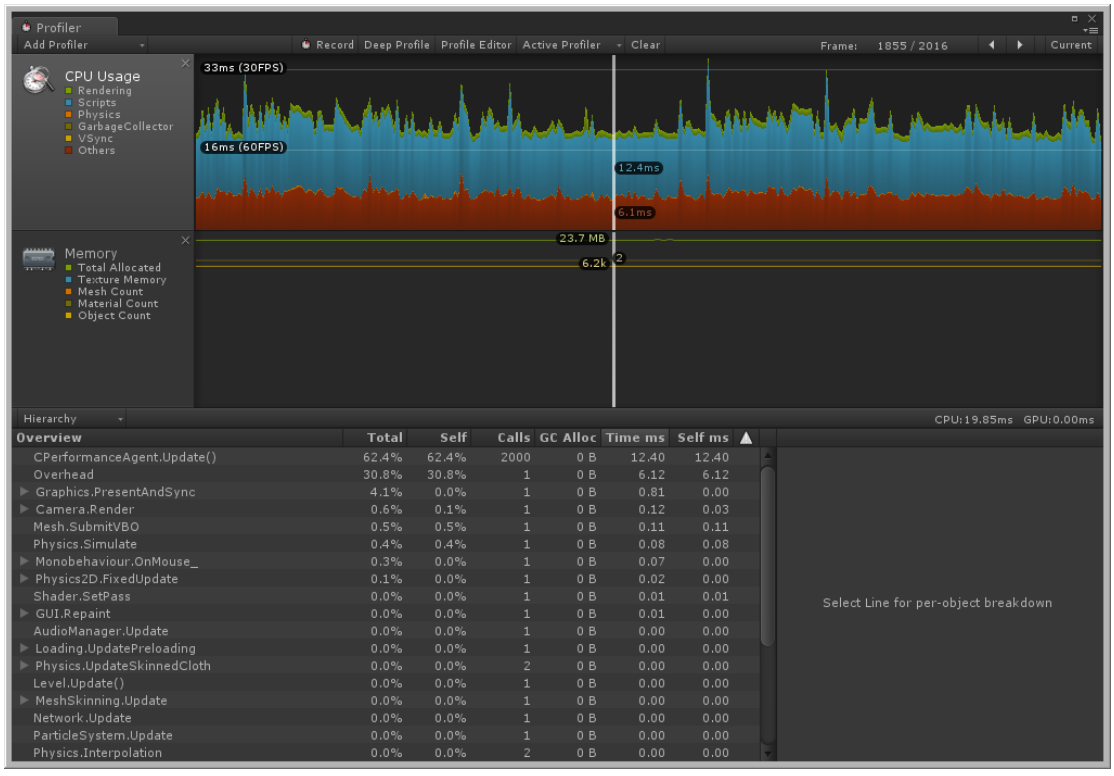


1000 Agents

https://github.com/TencentOpen/behaviac

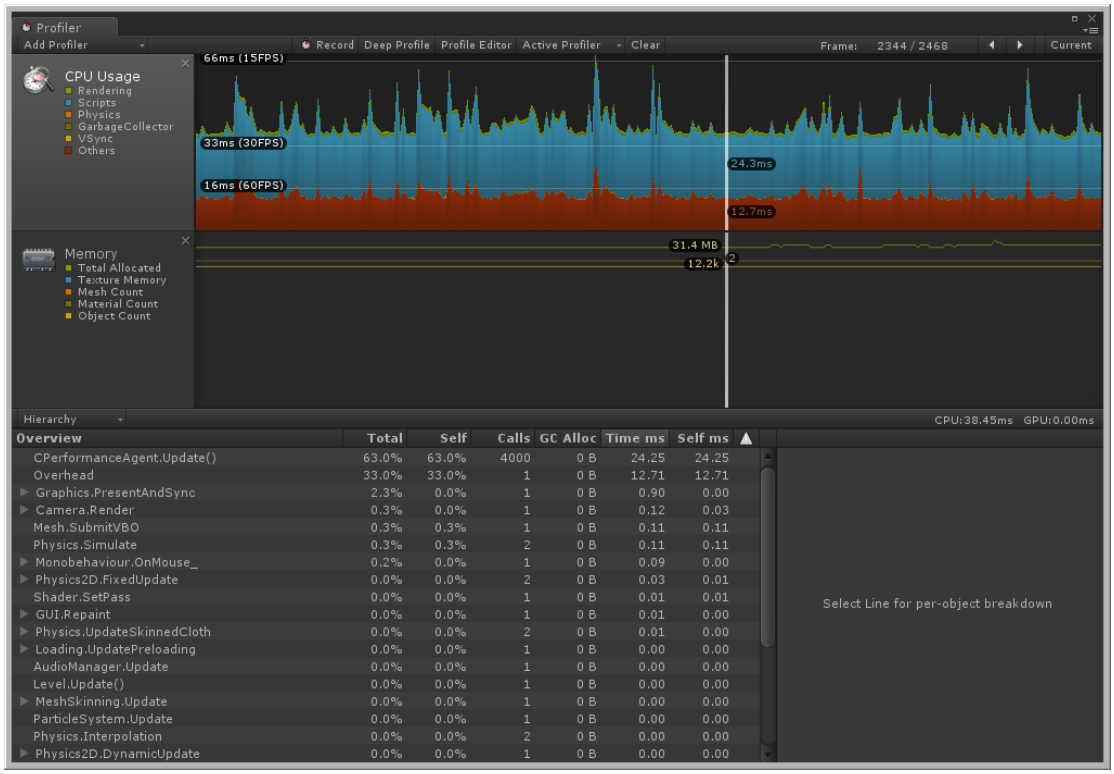


2000 Agents



https://github.com/TencentOpen/behaviac

4000 Agents



2.4 总计

2.4.1 PC，i7 2.93GHz, 8G RAM

Agents	1	10	100	1000	2000	4000
Total	0.01	0.04	0.37	2.21	3.65	4.97
Average	0.0100	0.0040	0.0037	0.0022	0.0018	0.0012

2.4.2 Android，MX5Q Dual Core 1.4GHz，1G RAM

Agents	1	10	100	1000	2000	4000
Total	0.04	0.41	1.92	12.66	24.88	49.91
Average	0.0400	0.0410	0.0192	0.0127	0.0124	0.0125

2.4.3 Android，MSM8974 Quad Core 2.2GHz，2G RAM

Agents	1	10	100	1000	2000	4000
Total	0.01	0.09	0.82	5.82	12.4	24.25

https://github.com/TencentOpen/behaviac

Average	0.0100	0.0090	0.0082	0.0058	0.0062	0.0061

3 Xml 格式和 CPP 格式效率的对比

用 xml 格式运行，分别 1,10,100 个 Agent 的结果：

	Total	Job	Overhead	Count	Overhead Avg
xml1	8.633967	8.595633	0.038334	14238	0.00000269
xml10	86.007768	85.645566	0.362202	142380	0.00000254
xml100	860.324909	856.518745	3.806164	1423800	0.00000267

用 cpp 格式运行，分别 1,10,100 个 Agent 的结果：

	Total	Job	Overhead	Count	Overhead Avg
cpp1	8.589980	8.569287	0.020693	14238	0.00000145
cpp10	86.073221	85.857756	0.215465	142380	0.00000151
cpp100	856.80929	854.694705	2.114584	1423800	0.00000149

由此可知 Cpp 格式的运行效率相比 xml 格式提升的是 50%不到一点。

cpp1/xml1	0.54
cpp10/xml10	0.59
cpp100/xml100	0.56

4 Xml 格式和 C#格式效率的对比

用 xml 格式运行，分别 1,10,100 个 Agent 的结果：

Agents	1	10	100
Total	0.02	0.13	0.61
Average	0.0200	0.0130	0.0061

用 c#格式运行，分别 1,10,100 个 Agent 的结果：

Agents	1	10	100
Total	0.01	0.04	0.37
Average	0.0100	0.0040	0.0037

由此可知 C#格式的运行效率相比 xml 格式提升的是 50%左右。

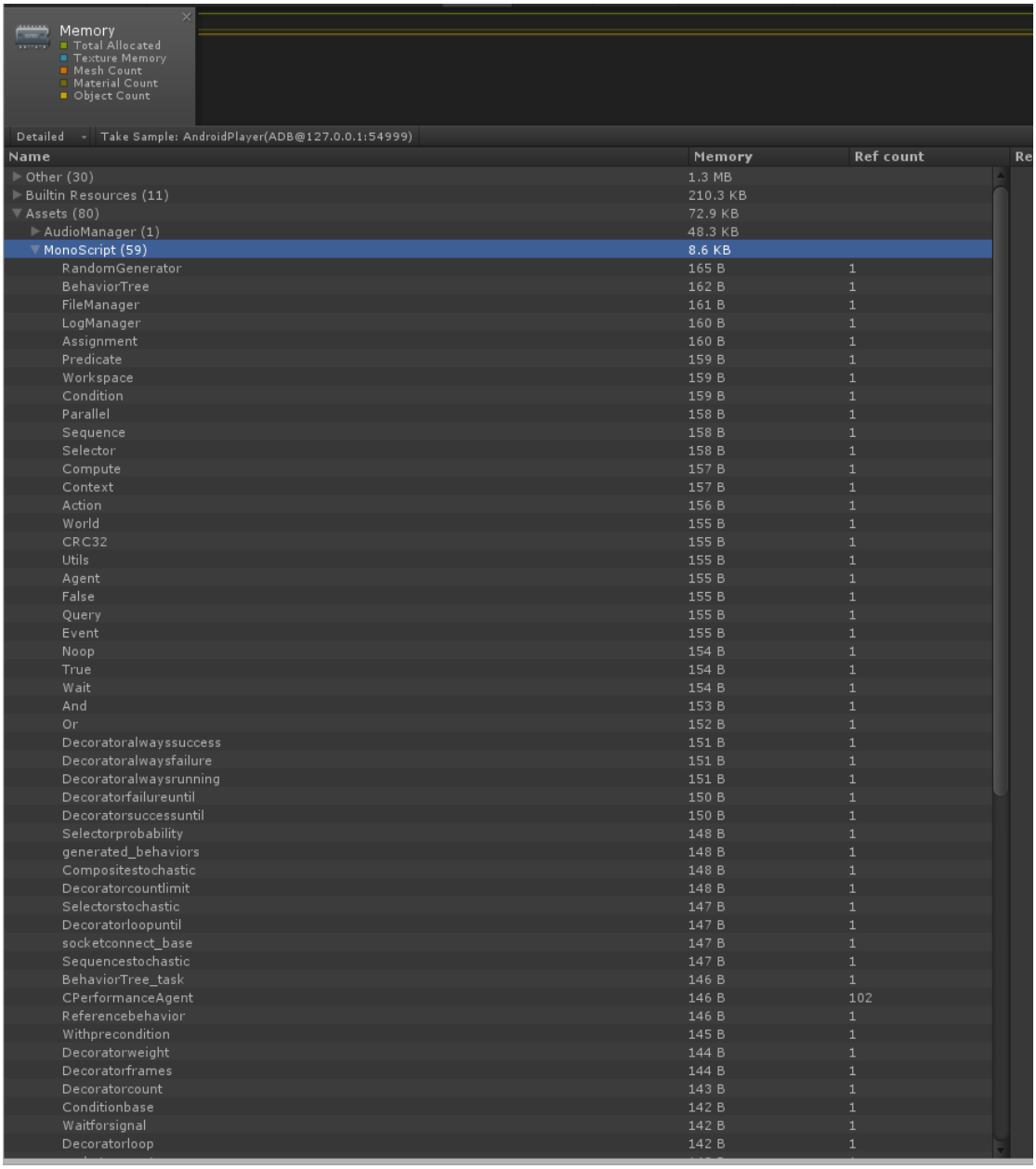
https://github.com/TencentOpen/behaviac

c#1/xml1	0.50
c#10/xml10	0.31
c#100/xml100	0.61

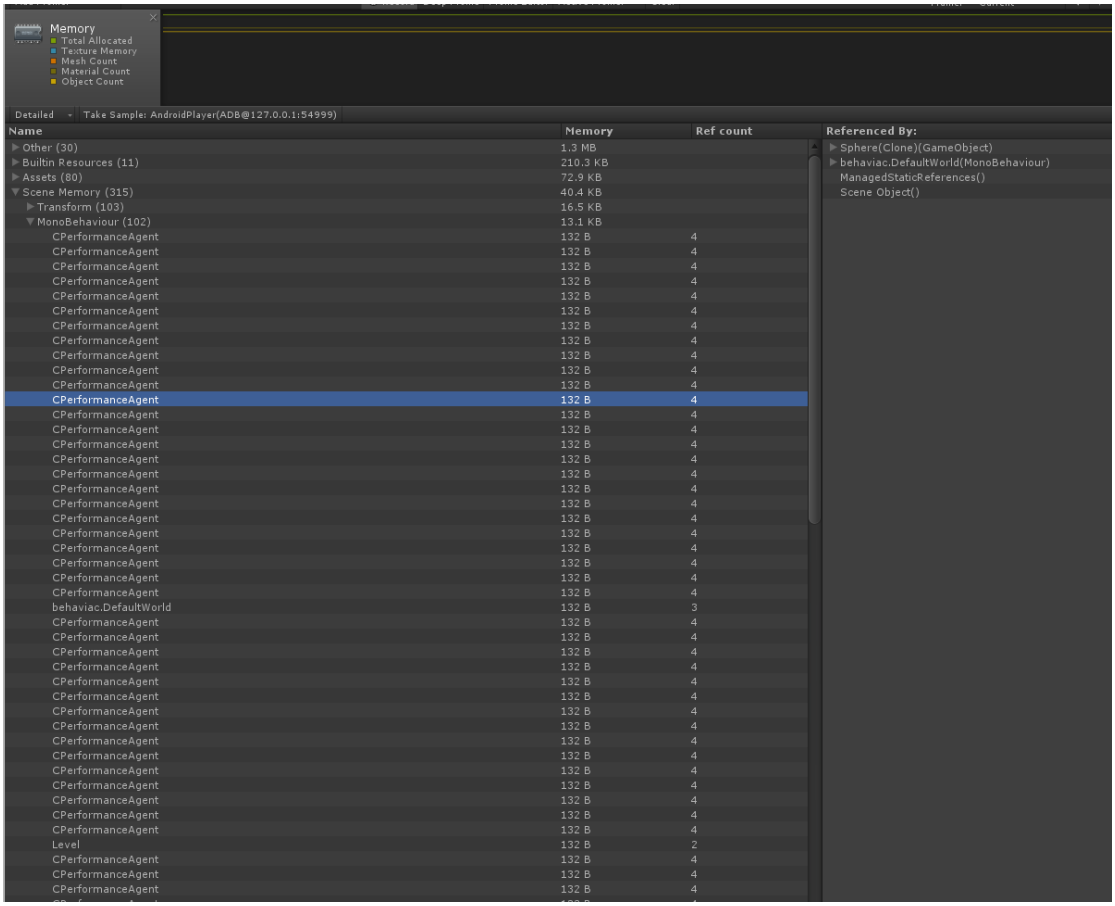
5 内存使用

下面两个图是内存测试的截图。该内存测试是在一个空的 scene 中在运行的的时候 Instantiate 出来 100 份 GameObject，每个 GameObject 只包含一个 CPerformanceAgent 的 script。

第一个图是所有相关 Script 所占内存，共 8.6KB，第二个图是运行时 MonoBehaviour 所占内存。



https://github.com/TencentOpen/behaviac



https://github.com/TencentOpen/behaviac